View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Elsevier - Publisher Connector

Information Processing Letters 114 (2014) 60–65

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

The firing squad synchronization problem with sub-generals

Kazuya Yamashita [a,*], Yasuaki Nishitani [b], Sadaki Hirose [a], Satoshi Okawa [c], Nobuyasu Osato [d]

[a] Faculty of Engineering, University of Toyama, Japan
[b] Faculty of Engineering, Iwate University, Japan
[c] Faculty of Computer Science and Engineering, University of Aizu, Japan
[d] Faculty of Information Science, Osaka Institute of Technology, Japan

ARTICLE INFO

Article history:
Received 5 October 2010
Received in revised form 7 August 2013
Accepted 8 August 2013
Available online 16 August 2013
Communicated by M. Yamashita

Keywords:
Algorithms
Firing squad synchronization problem
Cellular automaton

ABSTRACT

The *Firing Squad Synchronization Problem* (FSSP), one of the most well-known problems related to cellular automata, was originally proposed by Myhill in 1957 and became famous through the work of Moore. The first solution to this problem was given by Minsky and McCarthy, and a minimal time solution was given by Goto. A considerable amount of research has also dealt with variants of this problem. In this paper, we introduce a new state called the *sub-general* to the original problem and propose the FSSP with sub-generals. In particular, we consider the case of one sub-general and determine the position of the sub-general in the array that minimizes the synchronization time. Moreover, we determine the minimal time to solve this problem and show that there exists no minimal time solution for any length of array. However, we show that the total time of our algorithm approaches arbitrarily close to the minimal time.

© 2013 The Authors. Published by Elsevier B.V. Open access under CC BY-NC-SA license.

1. Introduction

The *Firing Squad Synchronization Problem* (FSSP), one of the most well-known problems related to cellular automata, was originally proposed by Myhill in 1957 and became famous through the work of Moore [1]. The first solution to the FSSP was given by Minsky and McCarthy [2], and requires $3n + O(\log n)$ steps for n cells using thirteen states. It is easy to show that any solution to the FSSP requires at least $2(n-1)$ steps. Goto [3] gave a minimal time solution using several thousands of states. Inspired by his results, many researchers tried to reduce the number of states: in 1966, Waksman [4] gave a solution with sixteen states; in 1967, Balzer [5] gave a solution with eight states; in 1987, Gerken [6] reduced this to seven states; and in the same year, Mazoyer [7] gave a solution with six states, the minimal time solution with the fewest states at present. Moreover, it has been shown [5] that any solution needs at least five states.[1]

A considerable amount of research has also dealt with variants of the FSSP. The FSSP has been studied on a ring of n cells [9], and on arrays of two and three dimensions [10,11]. The FSSP with multi-generals [12–14] has also been studied. The FSSPs for Cayley graphs [15] and another particular class of graphs [16] have also been studied. Some constrained variants of the FSSP have been developed to find solutions on reversible (i.e., backward deterministic) cellular automata [17] and cellular automata with a number-conserving property (i.e., a state is a tuple of non-negative integers whose sum is constant) [18]. Other kinds of constraints which have been considered involve the amount of information exchange between any pair of adjacent cells [19,20].

* Corresponding author.
E-mail address: kazuya@eng.u-toyama.ac.jp (K. Yamashita).

[1] While Balzer's statement is correct, his proof is unfortunately wrong. See page 101 of the paper by Sanders [8].

0020-0190 © 2013 The Authors. Published by Elsevier B.V. Open access under CC BY-NC-SA license.
http://dx.doi.org/10.1016/j.ipl.2013.08.003

In this paper, we introduce a new state called the *sub-general* to the original problem and propose the FSSP with sub-generals. In particular, we consider the case of one sub-general and determine the position of the sub-general in the array that minimizes the synchronization time. Moreover, we clarify the minimal time to solve this problem and show that no minimal time solution exists. However, we show that the total time of our algorithm approaches arbitrarily close to the minimal time.

2. The firing squad synchronization problem

Since the FSSP is defined by cellular automata, we explain them briefly here.

A cellular automaton is defined by an interconnection network of finite automata. Finite automata on nodes of the network, called *cells*, are copies of a finite automaton and communicate through edges of the network. For a cell v, cells connected to v directly are called *neighbor* cells of v. Each cell changes its state in discrete time by the state transition function depending on its own state and its neighbor cells' states (a set of a cell v and its neighbor cells is called its *neighborhood*). The interval of the time in which cells change their states once is called a *step*.

For a one-dimensional cellular automaton, an interconnection network is a path of length n for any positive integer n, and the neighbor cells of v are the ones immediately to its right and left. Therefore, this is called a three-neighborhood one-dimensional cellular automaton.

2.1. Definition of the FSSP

The FSSP is defined on three-neighborhood one-dimensional cellular automata.

Consider a one-dimensional array of n cells, $C_0, C_1, \ldots, C_{n-1}$. Assume that n is arbitrary, but finite. The cell C_0 at the left end is called a *general*, and the remaining cells $C_1, C_2, \ldots, C_{n-1}$ are called *soldiers*. The problem is to design a set of states and a state transition function so that the general can cause all cells to transit to a particular terminal state, called the *firing* state, for the first time and at exactly the same time.

A more precise definition is as follows. $\mathcal{A} = (\mathcal{S}, \delta)$ is the finite automaton of each cell C_i $(0 \leqslant i \leqslant n-1)$. $\mathcal{S}$ is a finite set of states: a *general* state $G \in \mathcal{S}$, a *quiescent* state $Q \in \mathcal{S}$, and a *firing* state $F \in \mathcal{S}$, each differing from the others. The state transition function is $\delta : (\mathcal{S} \cup \{B\}) \times \mathcal{S} \times (\mathcal{S} \cup \{B\}) \to \mathcal{S}$, where $B \notin \mathcal{S}$ is a signal that indicates the border of the array. For the quiescent state Q, we define $\delta(s_Q, Q, s_Q) = Q$, where s_Q indicates any element of $\{Q, B\}$.

Initially, all soldiers are assumed to be in the quiescent state, and only the general is assumed to be in the general state. The state of C_i $(0 \leqslant i \leqslant n-1)$ at time t is denoted by s_i^t. The state s_i^{t+1}, which is the state of C_i at time $t+1$, is determined by the state transition function and the states of its neighborhood at time t:

$$s_i^{t+1} = \delta\left(s_{i-1}^t, s_i^t, s_{i+1}^t\right).$$

If all cells first transit to the firing state F at time t_F, we say that the array of n cells is *fired* by the finite automaton $\mathcal{A}$. So, the sequence of states of each cell C_i $(0 \leqslant i \leqslant n-1)$ is

$$s_i^{t_F} = F \quad \text{and} \quad s_i^t \neq F \quad \text{for all } 0 \leqslant t < t_F.$$

A finite automaton $\mathcal{A}$ solving the FSSP for all n is called a *solution*. The time t_F required to solve the FSSP by a finite automaton $\mathcal{A}$ for an array of n cells is written as $t_F(n, \mathcal{A})$. The *minimal time* of the solutions for an array of n cells is written as $t_{F\,\min}(n)$, where

$$t_{F\,\min}(n) = \min\left\{t_F(n, \mathcal{A}) \;\middle|\; \mathcal{A} \text{ is a solution for the FSSP}\right\}.$$

If $t_F(n, \mathcal{A}) = t_{F\,\min}(n)$ holds for all n, the solution $\mathcal{A}$ is called a *minimal time solution*. In this case the solution $\mathcal{A}$ must achieve synchronization in minimal time for all n.

2.2. Outline of a minimal time solution of the FSSP

An outline of a minimal time solution (Waksman's algorithm [4]) of the FSSP is as follows.

At time $t = 0$, the general C_0 simultaneously sends signals,[2] at propagation speeds of $\frac{1}{1}, \frac{1}{3}, \frac{1}{7}, \ldots, \frac{1}{2^k-1}, \ldots$, where k is any positive integer[3] to the right. These signals are called the *firing signals* and play an important role in dividing the array into two, four, eight, ..., equal parts synchronously. When a signal with a propagation speed of $\frac{1}{1}$ reaches the right end of the array at time $t = n-1$, the rightmost cell C_{n-1} transits to the general state and behaves like the original general, except that it sends the firing signal to the left instead of the right. The signal with propagation speed $\frac{1}{3}$ which C_0 sent at time $t = 0$ and the signal with propagation speed $\frac{1}{1}$ which C_{n-1} sent at time $t = n-1$ meet at the center of the array, midway between C_0 and C_{n-1}. Then, the middle cell transits to the general state and sends firing signals to the left and the right. As a result of this process, the array is divided into two equal parts. This process is repeated until all cells in the array are in the general state. Every cell transits to the firing state at the next step. Note that the above process ensures synchronization of the array and also permits the determination of the firing time if a cell as well as the cells on either side of it are in the general state.

It is easily understood that the time required for the above-mentioned process is $2(n-1)$ steps. This equals the time required for the signal with propagation speed $\frac{1}{1}$, sent by C_0, to reach C_{n-1} and return.

Fig. 1 shows a time–space diagram of the minimal time algorithm. The horizontal axis is the cell space and the vertical axis is the time. The fractions in the figure are the propagation speeds of the firing signals and the dots indicate cells that are in the general state.

[2] A signal with a propagation speed of $\frac{1}{t}$ moves at a rate of one cell every t steps.

[3] This unbounded number of different signals can be realized with a finite number of states. See for example [7] for details.

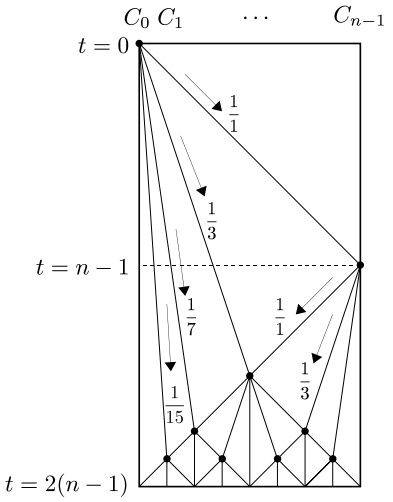

Fig. 1. Time–space diagram of Waksman's minimal time algorithm.

3. The FSSP with sub-generals

We introduce a new state G_S called the *sub-general* to the original problem and propose the FSSP with sub-generals.

The sub-general state G_S satisfies the following conditions where s_Q, s, and $*$ indicate any elements of $\{Q, B\}$, $\mathcal{S}$, and $\mathcal{S} \cup \{B\}$, respectively.

$$\delta(s_Q, G_S, s_Q) = G_S,$$

$$\delta(*, s, G_S) = \delta(*, s, Q),$$

$$\delta(G_S, s, *) = \delta(Q, s, *).$$

The sub-general state resembles the quiescent state: it does not change state while to its right and left is a border or cells in the quiescent state and it cannot be distinguished from the quiescent state by its neighbor cells. However, once it receives a signal (order) from the general, it can transit to a state different from the quiescent state and can even behave the same as the general. So, it is called the sub-general state.

Now, we define the FSSP with sub-generals. Let α be an arbitrary positive integer, M be a subset of $\{m \in \mathbb{N} \mid 0 < m < \alpha\}$.[4] We assume n equals $1 + l\alpha$ for a positive integer l. The initial configuration for the cells C_i $(0 \leqslant i \leqslant n-1)$ is denoted by $A_{\alpha,M,l}$ where the initial state of C_i is

$$s_i^0 = \begin{cases} G & \text{if } i = 0; \\ G_S & \text{if } i = lm,\ m \in M; \\ Q & \text{otherwise.} \end{cases}$$

Then the FSSP with sub-generals for a specific value α and a specific set M is the set of initial configurations

$$\mathrm{FSSP}_{\mathrm{SUB}}(\alpha, M) = \{A_{\alpha,M,l} \mid l \in \mathbb{N}\}.$$

Changing the value α or the set M defines a different algorithmic problem. To put it briefly, the state transition function knows the value α and the set M, but does not know the values n and l.

[4] $\mathbb{N}$ is the set of positive integers excluding 0.

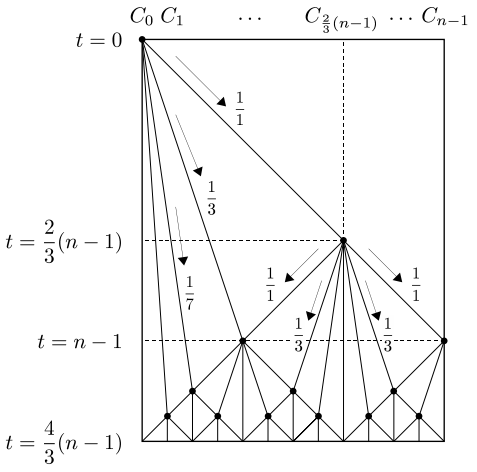

Fig. 2. Time–space diagram of Algorithm 1.

4. The firing squad synchronization algorithm

We consider the case of one sub-general and determine the position of the sub-general in the array that minimizes the synchronization time.

The cell $C_{ml} = C_{\frac{m}{\alpha}(n-1)}$ (where m is a positive integer that satisfies $0 < m < \alpha$) is the sub-general.

4.1. A simple algorithm

Intuitively, when Waksman's minimal time algorithm of the original FSSP is used, it seems to be the most efficient to place the sub-general at position of $\frac{2}{3}(n-1)$ from the left of the array. We give a simple algorithm to solve the FSSP with a sub-general when $\alpha = 3$ and $m = 2$.

Algorithm 1.

1. At time $t = 0$, the general C_0 sends the firing signals to the right.
2. At time $t = \frac{2}{3}(n-1)$, the sub-general $C_{2l} = C_{\frac{2}{3}(n-1)}$ receives the signal with propagation speed $\frac{1}{1}$ from the general and transits to the general state.
3. The sub-general, acting as a general, applies the minimal time algorithm to the right-hand part of array consisting of $C_{\frac{2}{3}(n-1)}, C_{\frac{2}{3}(n-1)+1}, \ldots, C_{n-1}$. At the same time, it also behaves as the right end cell of the segment of cells $C_0, C_1, \ldots, C_{\frac{2}{3}(n-1)}$.

It takes $\frac{4}{3}(n-1)$ steps to fire the array of length $\frac{2}{3}(n-1)+1$ to the left of the sub-general. It takes $\frac{2}{3}(n-1)$ steps for the sub-general to receive the signal with propagation speed $\frac{1}{1}$ from the general. Moreover, it takes $\frac{2}{3}(n-1)$ steps to fire the array of length $\frac{1}{3}(n-1)+1$ to the right of the sub-general. Therefore, when cell $C_{\frac{2}{3}(n-1)}$ is the sub-general for an array of length n, Algorithm 1 requires $\frac{4}{3}(n-1)$ steps.

Fig. 2 shows a time–space diagram of Algorithm 1.

4.2. Faster algorithms

We have given a simple algorithm which requires $\frac{4}{3}(n-1)$ steps by applying Waksman's minimal time algorithm. However, we can obtain faster algorithms by synchronizing the equally-spaced cells C_{jl} $(0 \leqslant j \leqslant \alpha)$.

First, we consider the case that the sub-general is placed in the right-hand half of the array, that is, $\frac{\alpha}{2} < m < \alpha$. In this case we can employ the following algorithm.

Algorithm 2.

1. At time $t = 0$, the general[5] C_0 sends the firing signals and also sends signals T_j at propagation speeds of $\frac{m+j}{2m-1}$ $(0 \leqslant j \leqslant \alpha - m)$.
2. At time $t = ml$, the sub-general C_{ml} receives the firing signal with propagation speed $\frac{1}{1}$. Then, the sub-general sends the firing signals to the left and signals S_j at propagation speeds of $\frac{j}{m-1}$ $(0 \leqslant j \leqslant \alpha - m)$ to the right; in particular the signal S_0 does not move.
3. At time $t = (2m-1)l$, the sub-general C_{ml} receives the signal T_0. At the same time, the cells $C_{(m+j)l}$ $(0 < j \leqslant \alpha - m)$ receive the signals T_j and S_j simultaneously and transit to the general state.[6] Then, the sub-general C_{ml} and the cells $C_{(m+j)l}$ apply the minimal time algorithm to the connected arrays of length $l+1$ (into which the original array has been evenly divided) that have cells in the sub-general or general state at both ends.

It takes $2ml$ steps to fire the array to the left of the sub-general, which is equal to the time to solve the original FSSP for an array of length $ml+1$.

For the array to the right of the sub-general, at time $t = (2m-1)l$, the sub-general C_{ml} receives the signal T_0 and the cells $C_{(m+j)l}$ $(0 < j \leqslant \alpha - m)$ receive the signals T_j and S_j simultaneously and transit to the general state. Moreover, it takes l steps to fire the arrays of length $l+1$ which have cells in the sub-general or general state at both ends. Therefore, it also takes $(2m-1)l + l = 2ml$ steps to fire the array to the right of the sub-general.

As a result, when cell $C_{ml} = C_{\frac{m}{\alpha}(n-1)}$ $(\frac{\alpha}{2} < m < \alpha)$ is the sub-general for an array of length n, Algorithm 2 requires $2ml = \frac{2m}{\alpha}(n-1)$ steps. This equals the time required for the signal with propagation speed $\frac{1}{1}$, which the general C_0 sent, to reach the sub-general C_{ml} and return. Therefore, the solution time of this algorithm is shortened as the value of m approaches $\frac{\alpha}{2}$, that is, as the sub-general approaches the center of the array.

For example, Fig. 3 shows the time–space diagram of Algorithm 2 when $\alpha = 4$ and $m = 3$. In the figure, a circle (○) indicates the general state, squares (□) indicate the sub-general states, and dots (●) indicate the general states of Waksman's algorithm.

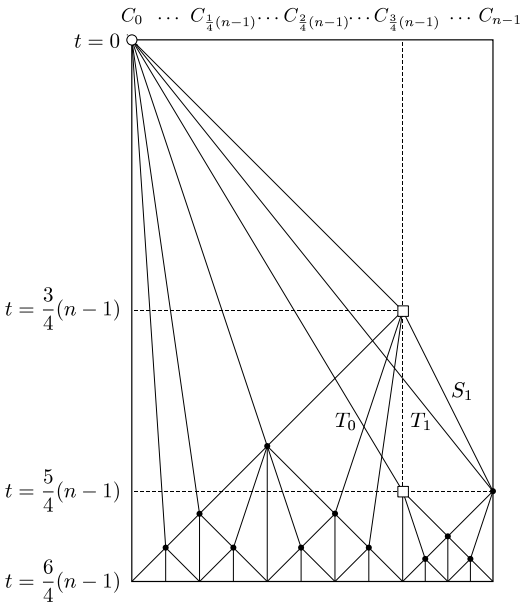

Fig. 3. Time–space diagram of Algorithm 2 when $\alpha = 4$ and $m = 3$.

Next we consider the case that the sub-general is placed in the left-hand half of the array, that is, $0 < m \leqslant \frac{\alpha}{2}$. In this case we can employ the following algorithm.

Algorithm 3.

1. At time $t = 0$, the general C_0 sends signals T_j at propagation speeds of $\frac{j}{\alpha}$ $(0 \leqslant j \leqslant \alpha)$.
2. At time $t = ml$, the sub-general C_{ml} receives the signal T_α. Then, the sub-general sends signals S_j at propagation speeds[7] of $\frac{j-m}{\alpha-m}$ $(0 \leqslant j \leqslant \alpha)$ to the right and the left.
3. At time $t = \alpha l = n-1$, the cell C_{n-1} at the right end receives the signal S_α and transits to the general state. At the same time, the cells C_{jl} $(0 \leqslant j \leqslant \alpha - 1)$ receive the signal T_j and S_j simultaneously and transit to the general state. The cells C_{jl} $(0 \leqslant j \leqslant \alpha)$ in the general state apply the minimal time algorithm to the connected arrays of length $l+1$ (into which the original array has been evenly divided) that have cells in the general state at both ends.

At time $t = n-1$, the cells C_{jl} $(0 \leqslant j \leqslant \alpha)$ transit to the general state simultaneously. Moreover, it takes l steps to fire the arrays of length $l+1$ which have cells in the general state at both ends. Therefore, when the cell C_{ml} $(0 < m \leqslant \frac{\alpha}{2})$ is a sub-general for an array of length n, Algorithm 3 requires $(n-1)+l = \frac{\alpha+1}{\alpha}(n-1)$ steps. The solution time of this algorithm does not depend on the position of the sub-general. By increasing the value of α, the solution time approaches arbitrarily close to $n-1$.

[5] Note that the general C_0 is in a different general state from that of Waksman's algorithm.

[6] Note that the general states at this time are the same as the general state of Waksman's algorithm.

[7] Note that a signal whose propagation speed is positive propagates to the right and a signal whose propagation speed is negative propagates to the left.

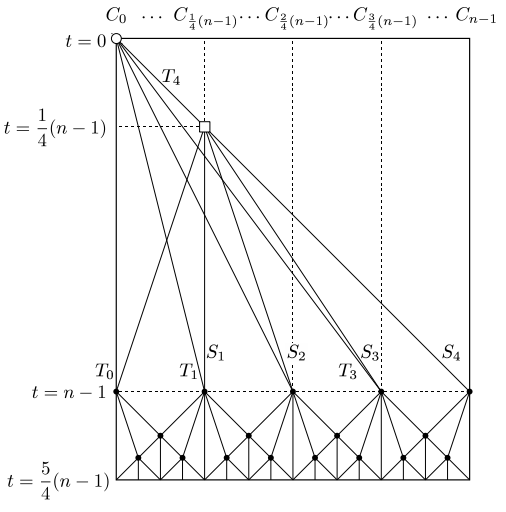

Fig. 4. Time–space diagram of Algorithm 3 when $\alpha = 4$ and $m = 1$.

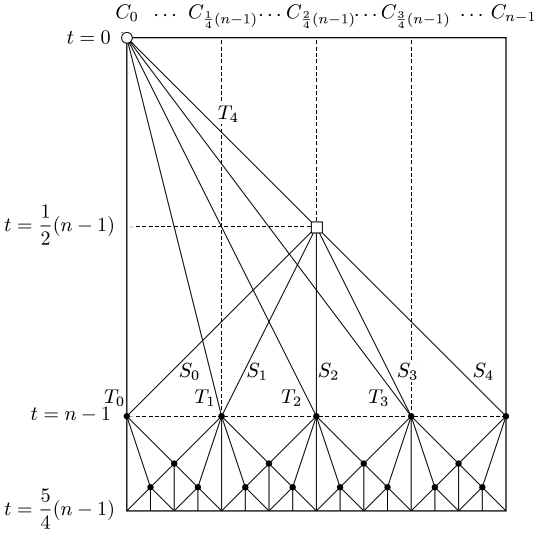

Fig. 5. Time–space diagram of Algorithm 3 when $\alpha = 4$ and $m = 2$.

For example, Figs. 4 and 5 show the time–space diagrams of Algorithm 3 when $\alpha = 4$, $m = 1$ and $\alpha = 4$, $m = 2$, respectively.

5. The minimal time to solve the FSSP with a sub-general

In this section, we determine the minimal time required to solve the FSSP with a sub-general for an array of length n.

Because cell C_{n-1} is in the quiescent state until time $t = n-2$, the solution time cannot be less than or equal to $n-2$. Therefore, the following lemma holds.

Lemma 1. *$t_F(n, \mathcal{A}) \geqslant n-1$ holds for any solution $\mathcal{A}$ of the FSSP with a sub-general for an array of length n.*

When cell $C_{ml_0} = C_{\frac{m}{\alpha}(n_0-1)}$ $(0 < m \leqslant \frac{\alpha}{2})$ is the sub-general for an array of a particular fixed length $n_0 = \alpha l_0 + 1$, we can easily construct a finite automaton $\mathcal{A}_{n_0}$ that can fire the array of length n_0 in $n_0 - 1$ steps but does not fire an array of length $n \neq n_0$. $\mathcal{A}_{n_0}$ consists of the following set of states $\mathcal{S}$ and state transitions where s and $*$ indicate any elements of $\mathcal{S} - \{X\}$ and $\mathcal{S} \cup \{B\}$, respectively.

$$\mathcal{S} = \{G = S_0, S_1, \ldots, S_{ml_0}, \ldots, S_{n_0-1} = F, Q, G_S, X\},$$

$$\delta(*, S_i, s) = S_{i+1},$$

$$\delta(S_{i-1}, Q, *) = S_i \quad \text{if } i \neq ml_0,$$

$$\delta(S_{ml_0-1}, G_S, *) = S_{ml_0},$$

$$\delta(S_{i-1}, G_S, *) = X \quad \text{if } i \neq ml_0,$$

$$\delta(S_{ml_0-1}, Q, *) = X,$$

$$\delta(*, S_i, X) = X,$$

$$\delta(X, Q, *) = Q,$$

$$\delta(X, Q_S, *) = Q_S.$$

We now give a brief description of the behavior of $\mathcal{A}_{n_0}$.

First, we consider the case $n = n_0$. The general C_0, initially in the state $G = S_0$, changes its state to $S_1, S_2, \ldots, S_{ml_0}, \ldots, S_{n_0-1}$ successively and the states propagate to the right one by one. As a result, at time $t = j$ $(0 < j \leqslant n_0 - 1)$, all cells from C_0 to C_j transit to the state S_j. So, at time $t = n_0 - 1$, all cells transit to the state $S_{n_0-1} = F$. Second, when the length $n < n_0$, the sub-general C_{ml} changes its state from G_S to X at time $t = ml$. The state X propagates to the left one by one. Therefore, at time $t = 2ml \leqslant n_0 - 1$, all cells are in the state X or Q. Similarly, when the length $n > n_0$, the cell C_{ml_0} changes its state from Q to X at time $t = ml_0$. The state X propagates to the left. Therefore, at time $t = 2ml_0 \leqslant n_0 - 1$, all cells are in the state X, Q, or G_S.

The finite automaton $\mathcal{A}'$ that mimics the above $\mathcal{A}_{n_0}$ together with an appropriate solution $\mathcal{A}$ is also a solution of the FSSP with a sub-general and $t_F(n_0, \mathcal{A}') = n_0 - 1$ holds.[8] Thus, the following theorem holds.

Theorem 1. *$t_{F\,\min}(n) = n - 1$ holds when the sub-general is placed to the left of the center of the array.*

6. The minimal time solution

Though we have shown that the minimal time to solve the FSSP with a sub-general for an array of length n is $n-1$, we must discuss whether such a minimal time solution exists or not.

The following lemma shows that no such solution exists.

Lemma 2. *Let $\mathcal{A} = (\mathcal{S}, \delta)$ be a solution of the FSSP with a sub-general for an array of length n. If $n - ml > |\mathcal{S}|^2$, $t_F(n, \mathcal{A}) > n - 1$ holds.*

[8] The states of the compound automaton $\mathcal{A}'$ consist of the firing state F and the pairs of states of two automata $\mathcal{A}_{n_0}$ and $\mathcal{A}$. The automaton $\mathcal{A}'$ simulates the two automata $\mathcal{A}_{n_0}$ and $\mathcal{A}$ simultaneously and transits to the firing state F when either of them fires.

Proof. We will prove the lemma by assuming $t_F(n, \mathcal{A}) = n-1$ and deriving a contradiction.

Let s_i^t be the state of the cell C_i at time t. If $n - ml > |\mathcal{S}|^2$, there exist two different integers i and j ($ml \leqslant i < j \leqslant n-2$) such that $s_{i-1}^i = s_{j-1}^j$, $s_i^i = s_j^j$. That is, there exist two pairs of adjacent cells $C_{i-1}C_i$ and $C_{j-1}C_j$ to the right of the sub-general C_{ml} for which the states $s_{i-1}^i s_i^i$ of the cells $C_{i-1}C_i$ at time i and the states $s_{j-1}^j s_j^j$ of the cells $C_{j-1}C_j$ at time j are the same.

Because $s_{i+1}^{i+1} = \delta(s_i^i, Q, Q)$ when $ml \leqslant i < n-2$, it follows that $s_{i+k}^{i+k} = s_{j+k}^{j+k}$ when $0 \leqslant k \leqslant n-2-j$. Furthermore, because $s_i^{i+1} = \delta(s_{i-1}^i, s_i^i, Q)$ when $ml < i \leqslant n-2$, it follows that $s_{i+k}^{i+k+1} = s_{j+k}^{j+k+1}$ when $0 \leqslant k \leqslant n-2-j$. Therefore, we have that $s_{n-2+i-j}^{n-1+i-j} = s_{n-2}^{n-1}$.

From the assumption that the solution time is $n-1$, $s_{n-2}^{n-1} = F$ holds. Therefore, $s_{n-2+i-j}^{n-1+i-j} = s_{n-2}^{n-1} = F$ holds. As the cell $C_{n-2+i-j}$ is in the firing state at time $t = n-1+i-j < n-1$, $\mathcal{A}$ is not the solution which contradicts the assumption. □

Thus we have the following theorem.

Theorem 2. *There is no minimal time solution for the FSSP with a sub-general.*

7. Conclusion

We introduced a new state called the sub-general to the original FSSP and proposed the FSSP with sub-generals. In particular, we considered the case of one sub-general.

When the sub-general is placed in the right-hand half of an array of length n, we gave an algorithm to solve the problem in $2ml$ steps. The solution time of this algorithm is shortened as the sub-general approaches the center of the array.

When the sub-general is placed in the left-hand half of an array of length n, we gave an algorithm that required $\frac{\alpha+1}{\alpha}(n-1)$ steps for a positive integer α. The solution time of this algorithm does not depend on the position of the sub-general. By increasing the value of α, we can approximate the solution time to $n-1$ steps.

Moreover, we determined that the minimal time to solve this problem is $n-1$ steps. Although we proved that no minimal time solution exists, we showed that the total time of our algorithm approaches arbitrarily close to the minimal time.

Acknowledgements

The authors would like to thank the editor and the reviewers who gave us valuable advice and suggestions.

References

[1] E.F. Moore, The firing squad synchronization problem, in: E.F. Moore (Ed.), Sequential Machines, Selected Papers, Addison-Wesley, Reading, MA, 1964, pp. 213–214.
[2] M. Minsky, J. McCarthy, Computation: Finite and Infinite Machines, Prentice-Hall, 1967, pp. 28–29.
[3] E. Goto, A minimal-time solution of the firing squad synchronization problem, in: Course Notes for Applied Mathematics, vol. 298, 1962, pp. 52–59.
[4] A. Waksman, An optimum solution to the firing squad synchronization problem, Inf. Control 9 (1966) 66–78.
[5] R. Balzer, An 8-state minimal time solution to the firing squad synchronization problem, Inf. Control 10 (1967) 22–42.
[6] H.-D. Gerken, Über Synchronisationsprobleme bei Zellularautomaten, Diplomarbeit, Technische Universität Braunschweig, 1987, pp. 1–50.
[7] J. Mazoyer, A six states minimal time solution to the firing squad synchronization problem, Theor. Comput. Sci. 50 (1987) 183–238.
[8] P. Sanders, Massively parallel search for transition-tables of polyautomata, in: Parcella '94, Akademie Verlag, 1994, pp. 99–108.
[9] K. Culik, Variations of the firing squad problem and applications, Inf. Process. Lett. 30 (1989) 153–157.
[10] I. Shinahr, Two and three-dimensional firing squad synchronization problems, Inf. Control 24 (1974) 163–180.
[11] K. Kobayashi, The firing squad synchronization problem for two dimensional arrays, Inf. Control 34 (1977) 153–157.
[12] M. Hisaoka, H. Yamada, M. Maeda, T. Worsch, H. Umeo, A design of firing squad synchronization algorithm for multi-general problems and their implementations, Tech. Rep. COMP 2002-133, IEICE, 2003.
[13] M. Hisaoka, H. Yamada, M. Maeda, T. Worsch, H. Umeo, An optimum-time firing squad synchronization algorithm for multi-general problem and its implementation, Tech. Rep. COMP 2003-18, IEICE, 2003.
[14] H. Schmid, T. Worsch, The firing squad synchronization problem with many generals for one-dimensional CA, in: IFIP TCS 2004, 2004, pp. 111–124.
[15] Z. Róka, The firing squad synchronization problem on Cayley graphs, in: Proc. of MFCS'95, Prague, Czech Republic, 1995, in: Lecture Notes in Computer Science, vol. 969, 1995, pp. 402–411.
[16] Y. Nishitani, N. Honda, The firing squad synchronization problem for graphs, Theor. Comput. Sci. 14 (1981) 39–61.
[17] K. Imai, K. Morita, Firing squad synchronization problem in reversible cellular automata, Theor. Comput. Sci. 165 (1996) 475–482.
[18] K. Imai, K. Morita, K. Sako, Firing squad synchronization problem in number-conserving cellular automata, in: Proc. of the IFIP Workshop on Cellular Automata, Santiago, Chile, 1998.
[19] J. Mazoyer, On optimal solutions to the firing squad synchronization problem, Theor. Comput. Sci. 168 (2) (1996) 367–404.
[20] H. Umeo, A design of cellular algorithms for 1-bit inter-cell communications and related cellular algorithms, in: Proc. of MCU'98 1, 1998, pp. 210–227.